CW1 : MD Mohidul Hasan , 15071717 , Group : (ESD 17)

Autonomous Reliable Car :

This report illustrates the basics of programming sensors and actuators of an Embedded platform employing basic programming methodologies of ISO C . The culmination of this project is to design and construct a smart car that can drive itself within a well-defined line and being able to stop when faced with an obstacle within range . Inside a mechatronic device , underlies a microcontroller architecture responsible for integration of intelligence , mechanical actions and sensory devices for a specified purpose (Mehmet Bodur, 2012) . For CW1 a distinct variety of sensors (Ultrasonic , IR sensor and infrared line sensors) having idiosyncratic functionalities have been used , the product being , an autonomous reliable car that can encapsulate data from its surrounding and analyse it through the use of sensors to avoid collision within an identified route employing actuators like electric motors .

Stage 1 : I/O Control :

Computer technology has seen incredible progression in the last 80 years since the first general-purpose electronic computer was invented . Advancements in the technology required to build computers and innovation in computer design has caused this massive progression (John L. Hennessy, 2006) . The foundations of computer architecture consist of a central processing unit (Control unit) , read and write memory , arithmetic logic unit (ALU) and miscellaneous input and output (I/O) devices like Keyboard, Graphics etc (Whisnant, 2005) .

Obstacle detection and avoidance is a primary goal of this project . There are two infrared sensors mounted at the top of the car chassis . Infrared sensors work on the principle of reflected light waves. Infrared light reflected from objects or sent from an infrared remote or beacon. Infrared sensors are also used to measure distance or proximity. The reflected light is detected and then an estimate of distance is calculated between sensor and object. The functions used to read the infrared sensors are initio_IrLeft (void) , initio_IrRight (void) and initio_IrAll (void) . These functions are declared as Boolean, so they only return a Boolean value of true(1) or false(0) and they don't take any parameters as arguments . The infrared sensor usually provides with a more accurate and consistent reading in distance measurement (CM). The autonomous car was vastly more responsive in obstacle avoidance when infrared sensor was used, compared to a sonar sensor although Infrared sensors can't work in dark environments while Ultrasonic Sensors can.

The sonar sensors are mostly used for object detection and avoidance and is essential for helping speed. However, it is not the best practice to collect the readings as it consists of a limited vision in its boundary. The line IR sensor provides different readings by detecting reflecting light coming from its own infrared, by measuring the reflected light shades it can detect from the lightest to the darkest (lines) shades. The autonomous car was providing relevant information to the actuators using the Boolean functions and expecting a desired outcome. For example, the robots have the capability to detect a black/dark line on a lighter surface depending on the contrast , they estimate whether the line underneath them is shifting towards their left/right as they move over them. Based on that estimation, they give respective signals to the motors to turn left/right to maintain a steady centre with respect to the line.

Sensors and actuators are two critical components of every closed loop control system. Such a system is also called a mechatronics system (M. Anjanappa).The actuators are the DC motors , coded to act upon the sensor values . The function used to control the DC motors are initio_DriveForward (speed) , initio_Stop () , void initio_TurnForward (leftSpeed, rightSpeed) and initio_TurnReverse (leftSpeed, rightSpeed) where some methods require speed(0 to 100) parameter as arguments while others require left and right speed value or no arguments at all .

The DC motors used to drive the robot car use pulse width modulation which allows them to have control of the rotational speed of the motors. Pulse width modulation control works by driving the DC motor with a series of on and off pulses. The amount of energy sent to the DC motor is provided by the parameter in the function initio_DriveForward (), a number between

1 and 100 is set energy to be sent to the DC motors. The energy sent is then delivered using pulse width modulation through a succession of pulses, this controls the rotational speed of the DC motors.
The electrical energy sent to the DC motors creates a torque which makes the motors spin, if the energy is too low, the motor will not move, because of the load and the friction between the wheels and the floor. If the energy is high, there will be less friction when spinning, therefore, the car will move. If the wheel is rotated in fast spinning pace around the vertical axis the robots roll down from the line, it causes friction, the force acts in the opposite direction to the way an object wants to slide.
Mvprint functions are analogous to printf() and it outputs the value
refresh () gives the actual output to the terminal, the refresh () routine is the identical, using stdscr as the default window.
initscr is normally the first curses routine to call when initializing a program.
The cbreak routine disables line buffering and erase character-processing (interrupt and flow control characters are unaffected), making characters typed by the user immediately available to the program.
The echo and noecho routines control whether characters typed by the user are echoed by getch as they are typed.

Stage 2 : Obstacle avoidance :

The functions used to allow the robot car to detect and avoid obstacles in front of it are initio_lrLeft () and initio_lrRight () . They use a Boolean data type to return whether there is an object in front or not. A logical OR operator is used to check both functions are returning 0. If one of the functions return a 1 it means an object has been detected by an IR sensor in which case the function initio_DriveForward (0) is used to obtain a safe mode . This function provides the parameter 0 which means no energy is sent to the motors and the robot stays still and does not move. If no object is detected by the IR sensors, then the function initio_DriveForward (100) is used. This provides the parameter 100 which means the maximum amount of energy is sent to the motors and the car carries on driving forward.

Stage 3 : Line Followers :

A Line following robot is an intelligent system as a kind of provisional one, which has robot positions corrective feedbacks, toward the black or white line (Marjani Bajestani, 2010) .The purpose of this course work is to build a line follower using the IR line function. The functions used to allow the robot to follow a well-defined black line surrounded by white lines are initio_IrLineLeft (void) and initio_IrLineRight (void) . The functions are declared as Boolean, so they return either true(1) or false(0) and the logical AND operator is used to check for the tests within the if statements of the void loop section of the code . If line sensor falls on black line it returns false(0) as the light from the sender is being absorbed by the black colour of the line . On the contrary if the sensor falls on white line it returns true(1) as the light from the sender is reflected to the receiver of the line sensor module . The actuators are employing the following functions : initio_Stop () , initio_DriveForward(speed), initio_DriveReverse (speed) , initio_SpinLeft (speed) , initio_SpinRight(speed) , initio_TurnForward (leftSpeed, rightSpeed) **and** initio_TurnReverse (leftSpeed, rightSpeed) based on the line IR sensors readings . If both line sensor readings are true, then the motors run the drive forward function with any value from 0 to 100 as argument for it. The functions spin left and right are employed based on the IR line sensor data reading being either 1 or 0. When the line sensors lose the line that is the sensor reading become 0 for both sensors at the same time , the motor is programmed to run the drive reverse function that takes any value from 0 to a 100 as an argument .
Control logic is a form of software adaption that takes control of operational tasks, after realising that there is no requirement for such intelligence within the task it became apparent that control logic was not used. Due to there being the need of intelligence in the task it was apparent that there was no use of control logic. Our forward speed was 75 so that we can get best performance outcome. In count, the drive reverse function (initio_DriveReverse) is the only thing that have been provided as extra everything else was provided with the skeleton code by our Supervisor.

CW1 : MD Mohidul Hasan , 15071717 , Group : (ESD 17)
Appendix A :

Test 1

LAP	Time(Seconds)	Comments
1	25.52	Driver Speed :70 Spin Left :100 Spin right :100 Average time :26.24
2	24.99	
3	26.75	
4	27.59	
5	26.34	

Test 2

LAP	Time(Seconds)	Comments
1	24.80	Driver Speed :72 Spin Left :100 Spin right :100 Average time :26.0
2	25.50	
3	26.39	
4	25.38	
5	27.96	

Test 3

LAP	Time(Seconds)	Comments
1	23.39	Driver Speed :75 Spin Left :100 Spin right :100 Average time :24.434
2	22.25	
3	26.96	
4	25.45	
5	24.12	

Test 4

LAP	Time(Seconds)	Comments
1	27.89	Driver Speed :65 Spin Left :100 Spin right :100 Average time :29.31
2	31.30	
3	29.35	
4	28.45	
5	29.56	

Test 5

LAP	Time(Seconds)	Comments
1	20.2	Driver Speed :80 Spin Left :100 Spin right :100 Average time :22.39
2	23.36	
3	24.23	
4	21.16	
5	22.99	

CW1 : MD Mohidul Hasan , 15071717 , Group : (ESD 17)

Appendix B : Line follower source code

```
//========================================================================
// author: Raimund Kirner, University of Hertfordshire
//        initial version: Oct.2016
//
// license: GNU LESSER GENERAL PUBLIC LICENSE
//         Version 2.1, February 1999
//         (for details see LICENSE file)
//
// Compilation:
// gcc -o testIR -Wall -Werror -lcurses -lwiringPi -lpthread -linitio testIR.c
//
//========================================================================

#include <stdlib.h>
#include <initio.h>
#include <curses.h>



//========================================================================
// testIR():
// Simple program to test infrared obstacle sensors and line follower sensors:
// Drive forward and stop whenever an obstacle is detected by either
// the left or right infrared (IR) sensor.
//========================================================================
void testIR(int argc, char *argv[])
{
 int ch = 0;
 int irL, irR ;
```

```
  int lfL, lfR ;  // white .. 1, black ... 0
while (ch != 'q') {
   mvprintw(1, 1,"%s: Press 'q' to end program", argv[0]);
   irL = initio_IrLeft();
   irR = initio_IrRight();
   lfL = initio_IrLineLeft(); // white .. 1, black ... 0
   lfR = initio_IrLineRight(); //

 if (irL != 0 || irR != 0) {
     mvprintw(3, 1,"Action: Stop (IR sensors: %d, %d)     ", irL, irR);
     initio_DriveForward (0); // Stop
   }
   // no obstacle ahead, so focus on line following
 else if ((lfL == 0) && (lfR == 0 )) {
     mvprintw(3, 1,"Action: Straight (Line sensors: %d, %d)    ", lfL, lfR);
     initio_DriveForward (80); // move straight forward
   }
   else if ((lfL == 1) && (lfR ==0 )) {
     // car is too much on the left
     mvprintw(3, 1,"Action: Spin left (Line sensors: %d, %d)    ", lfL, lfR);
     initio_SpinRight(100); // move towards left
   }
   else if ((lfR == 1) && (lfL == 0 )) {
     // car is too much on the right
     mvprintw(3, 1,"Action: Spin right (Line sensors: %d, %d)    ", lfL, lfR);
    initio_SpinLeft(90); //move towards right
}
   else
   {
     mvprintw(3, 1,"Lost my line (Line sensors: %d, %d)       ", lfL, lfR);
    initio_DriveReverse (100); // REFINEMENT OF LINE FOLLOWER ALGORITHM
```

```
}
   ch = getch();
   if (ch != ERR)
     mvprintw(2, 1,"Key code: '%c' (%d)", ch, ch);
   refresh();
  } // while
  return;
}



//========================================================================
// main(): initialisation of libraries, etc
//========================================================================
int main (int argc, char *argv[])
{
  WINDOW *mainwin = initscr();  // curses: init screen
  noecho ();                    // curses: prevent the key being echoed
  cbreak();                    // curses: disable line buffering
  nodelay(mainwin, TRUE);       // curses: set getch() as non-blocking
  keypad (mainwin, TRUE);       // curses: enable detection of cursor and other keys

  initio_Init (); // initio: init the library


  testIR(argc, argv);


  initio_Cleanup ();  // initio: cleanup the library (reset robot car)
  endwin();           // curses: cleanup the library
  return EXIT_SUCCESS;
}
```

John L. Hennessy, D. A. (2006). *Computer Architecture : A Quantitative Approach.* Retrieved November 22, 2018, from https://doc.lagout.org/Computer%20Architechture.pdf

M. Anjanappa, K. D. (n.d.). Introduction to Sensors and Actuators. Retrieved November 22, 2018, from http://www.kelm.ftn.uns.ac.rs/literatura/mur/IntroductionToSensorsAndActuators.pdf

Marjani Bajestani, S. E. (2010). *Technical report of building a line follower robot.* International Conference on Electronics and Information Engineering (ICEIE 2010). 1. V1-5. 10.1109/ICEIE.2010.5559826. Retrieved November 22, 2018, from https://www.researchgate.net/publication/251948999_Technical_report_of_building_a_line_follower_robot

Mehmet Bodur, A. P. (2012). C Programming of Microcontrollers for Hobby Robotics. Retrieved November 22, 2018, from https://www.researchgate.net/publication/266268853_C_Programming_of_Microcontrollers_for_Hobby_Robotics

Whisnant, J. M. (2005). *Foundations of Computer Architecture.* Stanford university. Retrieved November 22, 2018, from https://apps.ep.jhu.edu/course-homepages/2891-605.611-foundations-of-computer-architecture-whisnant